Enabling Continuous Improvement of a Continuous Integration Process

Vassallo, Carmine

Abstract: Continuous Integration (CI) is a widely-adopted software engineering practice. Despite its undisputed benefits, like higher software quality and improved developer productivity, mastering CI is not easy. Among the several barriers when transitioning to CI, developers need to face a new type of software failures (i.e., build failures) that requires them to understand complex build logs. Even when a team has successfully introduced a CI culture, living up to its principles and improving the CI practice are also challenging. In my research, I want to provide developers with the right support for establishing CI and the proper recommendations for continuously improving their CI process.

DOI: https://doi.org/10.1109/ASE.2019.00151

for the detection of CI anti-patterns [22] can be used to help developers understanding when they are not using CI properly. Recent work also investigated configuration smells both in the form of misuse of CI features [7] and security issues [14]. Our CI anti-patterns have different focus because we look at process-related smells rather than configuration issues.

Previous researchers have mainly studied the impact of using CI on source code quality [11] and developers' productivity [21], [26]. However, they neither provide a way to continuously assess the impact of CI in development teams nor recommend developers on how to improve their CI practice.

VII. Expected Contributions and Outlook

I made the following contributions at this point in my Ph.D.

- A taxonomy of build failures [25].
- A summarization approach that significantly improves the understandability of build failures [23].
- Definition of anti-patterns in CI (the corresponding work is currently under review at EMSE) and an approach for detecting their presence during project evolution [22].

I expect the following contributions for the rest of my Ph.D.

- Definition of a model for measuring the CI performance.
- Recommendation approach for suggesting practices that can improve a CI process.

The research questions **RQ 1-3** present the three milestones of my Ph.D. To reach the first milestone, we first built a taxonomy of build failure types (**RQ 1a**) that was the core contribution of our work published at ICSME '17 [25]. Then we implemented our Build Abstraction and Recovery Tool (BART) and we validated the effectiveness of our build summarization approach in a study conducted with professional developers. We published our approach to build failure resolution (**RQ 1b**) and the case-study results in a research paper accepted at ICPC '18 [23] and extended at EMSE [24]. The second milestone includes two works that were submitted to EMSE and ICSE respectively. The first work is about the catalog of CI anti-patterns (**RQ 2a**) and is currently under review. The second is already published in the proceedings of ICSE '19 [22] and answers **RQ 2b**. In the last milestone, I plan to answer **RQ 3** through two research papers. The first work will be about building a model for measuring the CI performance (**RQ 3a**). The second work will be about the recommendation of practices for improving CI (**RQ 3b**). I plan to submit them to top-level software engineering conferences by the middle of 2020 and defend my Ph.D. thesis by the end of next year.

VIII. Acknowledgments

I would like to thank Sebastian Proksch and Harald C. Gall for their supervision and support. This research received funding from the Swiss National Science Foundation (SNF) under project SURF-MobileAppsData (no. 200021-166275).

References

[1] Jenkins. https://jenkins.io (last access 19.06.2019).
[2] SonarQube. http://www.sonarqube.org (last access 19.06.2019).
[3] M. Beller, G. Gousios, and A. Zaidman. Oops, my tests broke the b An explorative analysis of Travis CI with GitHub. In *Internati Conference on Mining Software Repositories*, 2017.
[4] J. Cohen. A coefficient of agreement for nominal scales. *Educati and psychological measurement*, 20(1):37–46, 1960.
[5] P. Duvall, S. M. Matyas, and A. Glover. *Continuous Integra Improving Software Quality and Reducing Risk*. Addison-Wesley, 2
[6] N. Forsgren, J. Humble, and G. Kim. *Accelerate: The Science of I Software and DevOps Building and Scaling High Performing Techno Organizations*. IT Revolution Press, 1st edition, 2018.
[7] K. Gallaba and S. McIntosh. Use and misuse of continuous integra features: An empirical study of projects that (mis)use Travis CI. *I Transactions on Software Engineering*, (to appear):1, 2018.
[8] M. Hilton, N. Nelson, T. Tunnell, D. Marinov, and D. Dig. Tr offs in continuous integration: Assurance, security, and flexibility *Proceedings of the 25th ACM SIGSOFT International Symposiun Foundations of Software Engineering, FSE 2017*, 2017.
[9] M. Hilton, T. Tunnell, K. Huang, D. Marinov, and D. Dig. Us costs, and benefits of continuous integration in open-source proj In *Proceedings of the 31st IEEE/ACM International Conference Automated Software Engineering (ASE)*, 2016.
[10] N. Kerzazi, F. Khomh, and B. Adams. Why do automated builds br an empirical study. In *30th IEEE International Conference on Softv Maintenance and Evolution (ICSME)*. IEEE, 2014.
[11] F. Khomh, B. Adams, T. Dhaliwal, and Y. Zou. Understanding the im of rapid releases on software quality - the case of firefox. *Empi Software Engineering*, 20(2):336–373, 2015.
[12] C. Macho, S. McIntosh, and M. Pinzger. Automatically repai dependency-related build breakage. In *International Conference Software Analysis, Evolution, and Reengineering (SANER)*, 2018.
[13] D. Parmenter. *Key Performance Indicators: Developing, Implemen and Using Winning Kpis*. John Wiley & Sons, Inc., 2007.
[14] A. Rahman, C. Parnin, and L. Williams. The seven sins: Security sn in infrastructure as code scripts. In *41st International Conferenc Software Engineering (ICSE)*. IEEE/ACM, 2019.
[15] T. Rausch, W. Hummer, P. Leitner, and S. Schulte. An empirical ana of build failures in the continuous integration workflows of java-b open-source software. In *International Conference on Mining Softv Repositories (MSR)*. ACM, 2017.
[16] H. Seo, C. Sadowski, S. G. Elbaum, E. Aftandilian, and R. W. Bowd Programmers' build errors: a case study (at Google). In *Internati Conference on Software Engineering (ICSE)*, 2014.
[17] D. Spencer. *Card sorting: Designing usable categories*. Rosenfeld M 2009.
[18] StackOverflow. https://stackoverflow.com/. Accessed: 2019-06-19.
[19] D. Ståhl and J. Bosch. Automated Software Integration Flows in Indu A Multiple-case Study. In *ICSE*, 2014.
[20] S. Urli, Z. Yu, L. Seinturier, and M. Monperrus. How to design a prog repair bot?: Insights from the repairnator project. In *Proceeding the 40th International Conference on Software Engineering: Softv Engineering in Practice*, ICSE-SEIP '18. ACM, 2018.
[21] B. Vasilescu, Y. Yu, H. Wang, P. T. Devanbu, and V. Filkov. Quality productivity outcomes relating to continuous integration in github *ESEC/SIGSOFT FSE*, pages 805–816. ACM, 2015.
[22] C. Vassallo, S. Proksch, H. C. Gall, and M. Di Penta. Automated repo of anti-patterns and decay in continuous integration. In *Internati Conference on Software Engineering (ICSE)*. IEEE/ACM, 2019.
[23] C. Vassallo, S. Proksch, T. Zemp, and H. C. Gall. Un-break my b Assisting developers with build repair hints. In *International Confer on Program Comprehension (ICPC)*. IEEE, 2018.
[24] C. Vassallo, S. Proksch, T. Zemp, and H. C. Gall. Every build you b Developer-oriented assistance for build failure resolution. *Empi Software Engineering*, 2019. (To appear).
[25] C. Vassallo, G. Schermann, F. Zampetti, D. Romano, P. Lei A. Zaidman, M. D. Penta, and S. Panichella. A tale of CI build fail An open source and a financial organization perspective. In *ICS* pages 183–193. IEEE Computer Society, 2017.
[26] Y. Zhao, A. Serebrenik, Y. Zhou, V. Filkov, and B. Vasilescu. The im of continuous integration on other software development practice large-scale empirical study. In *IEEE/ACM International Conferenc Automated Software Engineering (ASE)*, 2017.